Assignment 2: Unity maze shortest path using (UCS, BFS, A\*, DFS)

Team Members: Driss Jaidi, Mohammed Chaouni

**Colors used in each path:**

**Blue color: DFS**

**Cyan color: A\***

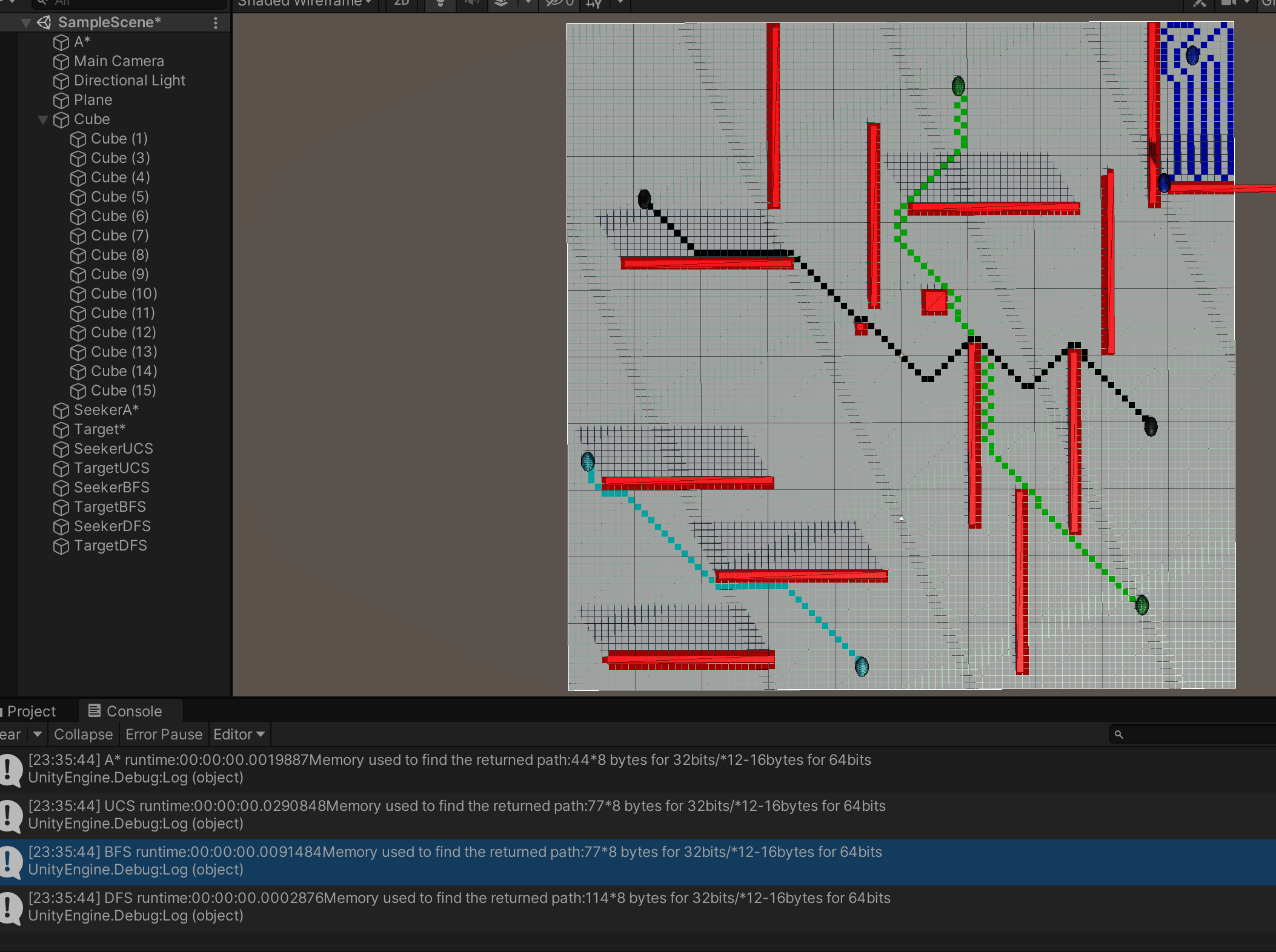
**Black Color: BFS**

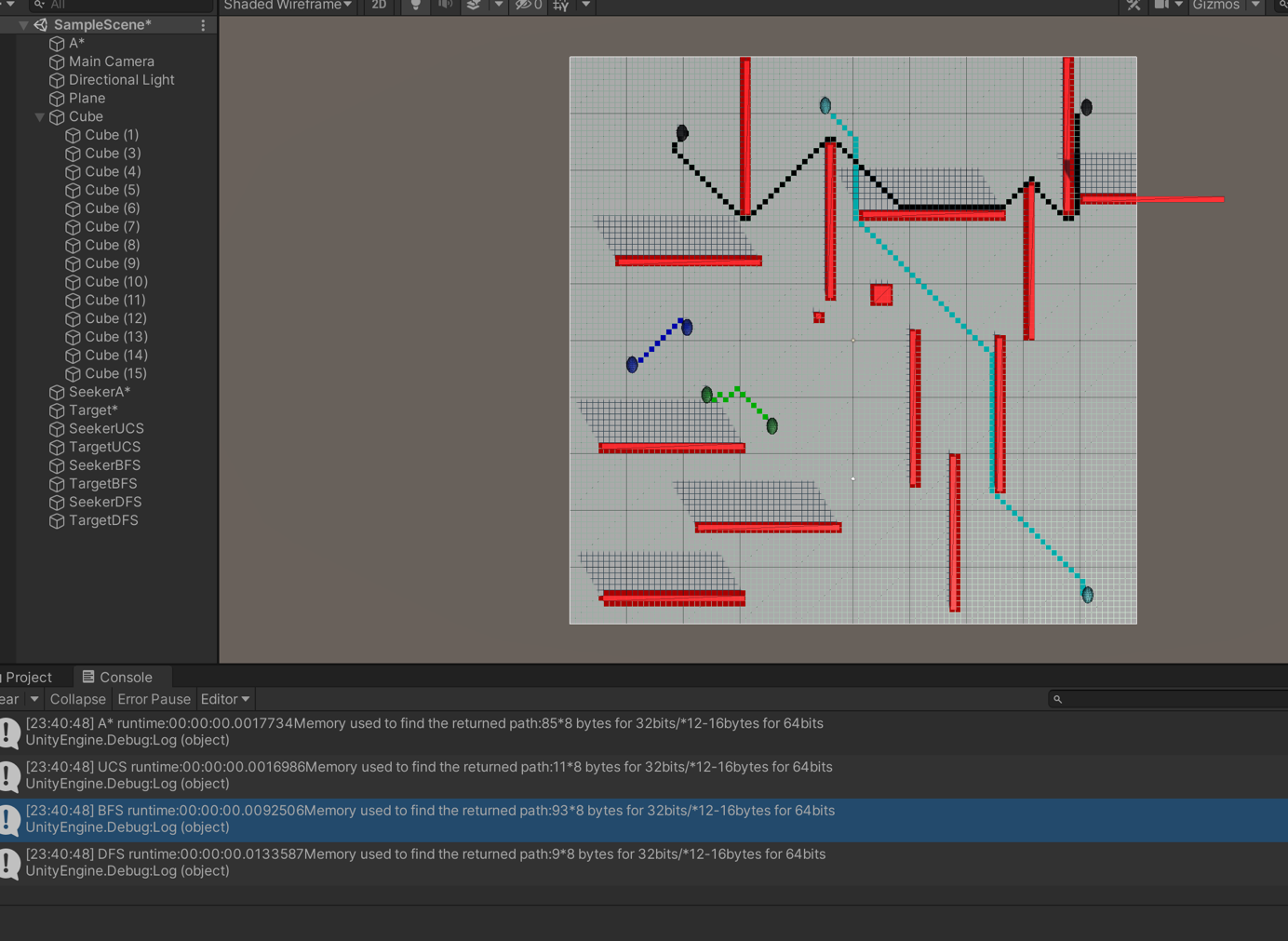
**Green color: UCS**

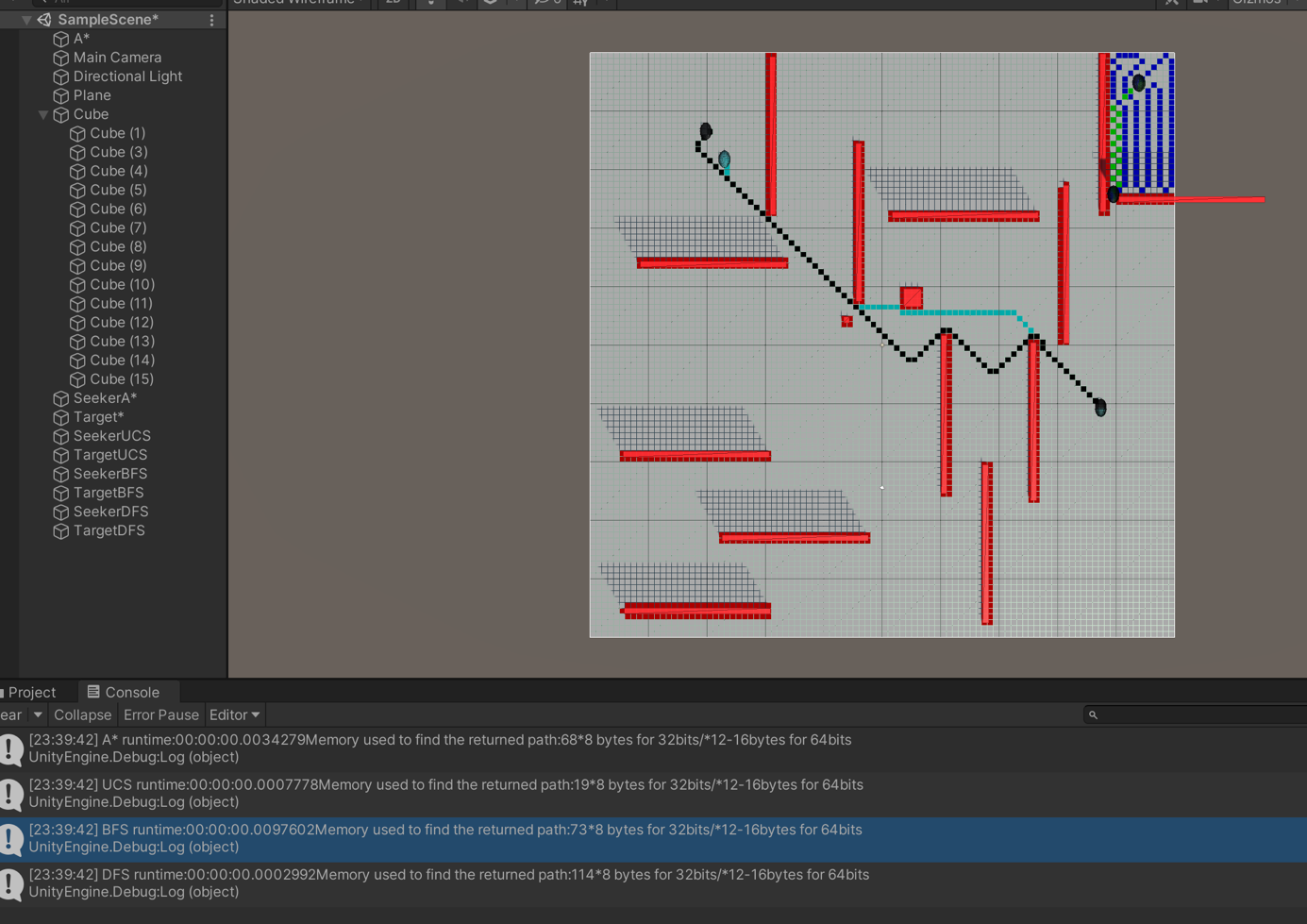
Credits, references, and mains functions/files

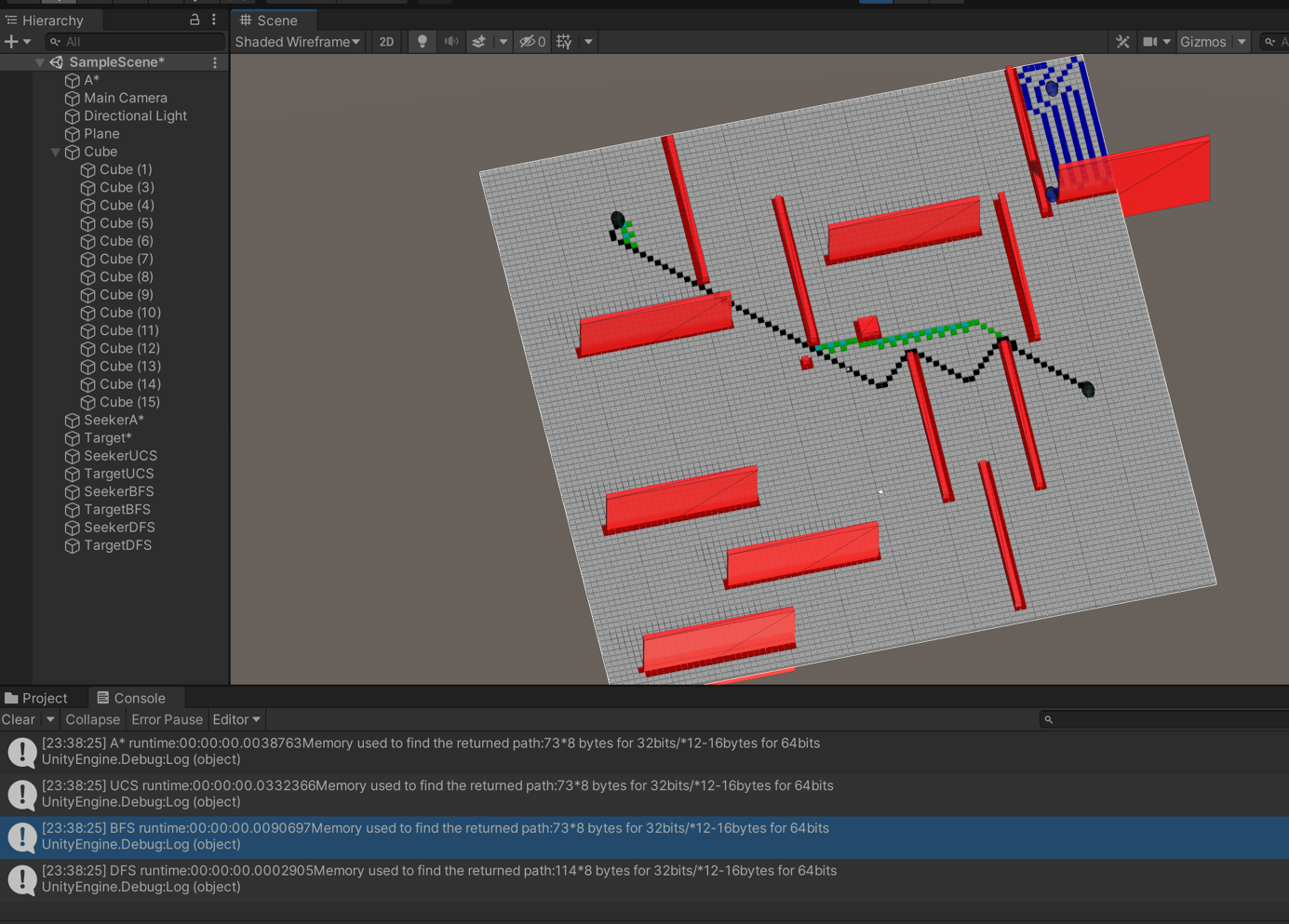
|  |
| --- |
|  |
|  |
|  | **Main File containing algorithms : Pathfinding.cs** |
|  | **Created by : Driss Jaidi and Mohammed Chaouni** |
|  | **Version # : 4.0** |
|  | **Updated on : October 14, 2021** |
|  | **References: { Sebastian Lague Youtube channel and github code** |
|  | **https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt\_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW** |
|  | **https://github.com/SebLague/Pathfinding** |
|  | **}** |
|  | **This program contains 10 functions:** |
|  | **FindPath: finds shortest path using A\*, the code can be modified for different A\* heuristics** |
|  | **FindPathUCS: finds shortest path using uniform cost search, the code can be modified for different gCosts** |
|  | **FindPathBFS: finds shortest path using Breadth first search.** |
|  | **FindPathDFS: finds shortest path using Depth first search.** |
|  | **RetracePath: marks the nodes that constitue the shortest path of A\*** |
|  | **RetracePathUCS: marks the nodes that constitue the shortest path of A\*** |
|  | **RetracePathBFS: marks the nodes that constitue the shortest path of A\*** |
|  | **RetracePathDFS: marks the nodes that constitue the shortest path of A\*** |
|  | **GetDistance: gets the optimal distance between the two parameters of the function considering that some directions have a higher cost that normal movement** |
|  | **GetDistanceUCS: gets the optimal distance between the two parameters of the function considering a straight line distance using sqrt(par1, par2) all to the power two**  **File containing colorisation and grid structure: Grid.cs**  **File containing the node structure: Node.cs**  **File containing the scenes from screenshots: scenes** |
|  |  |
|  |  |

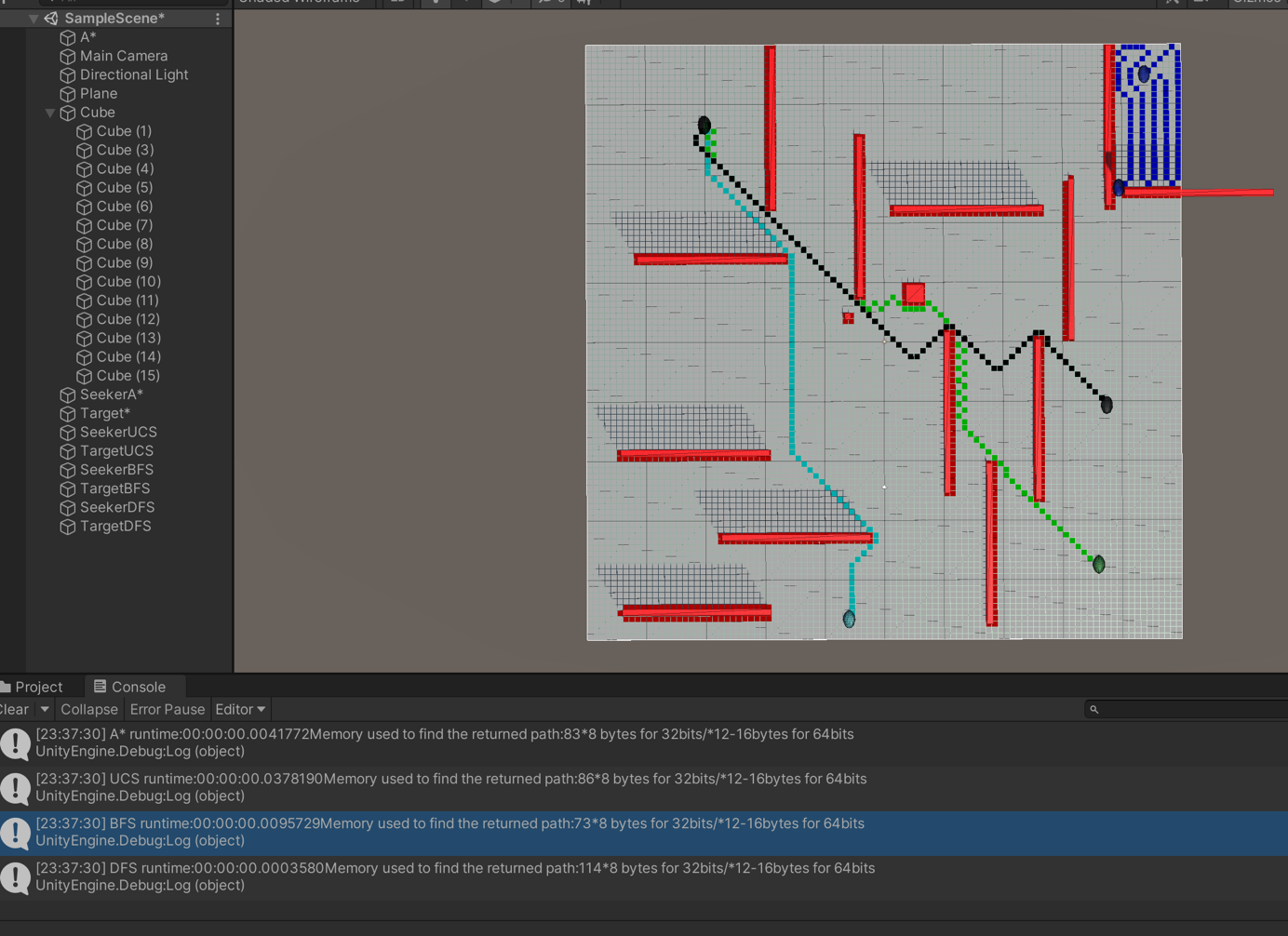
Snapshot1: A\* using the videos heuristics

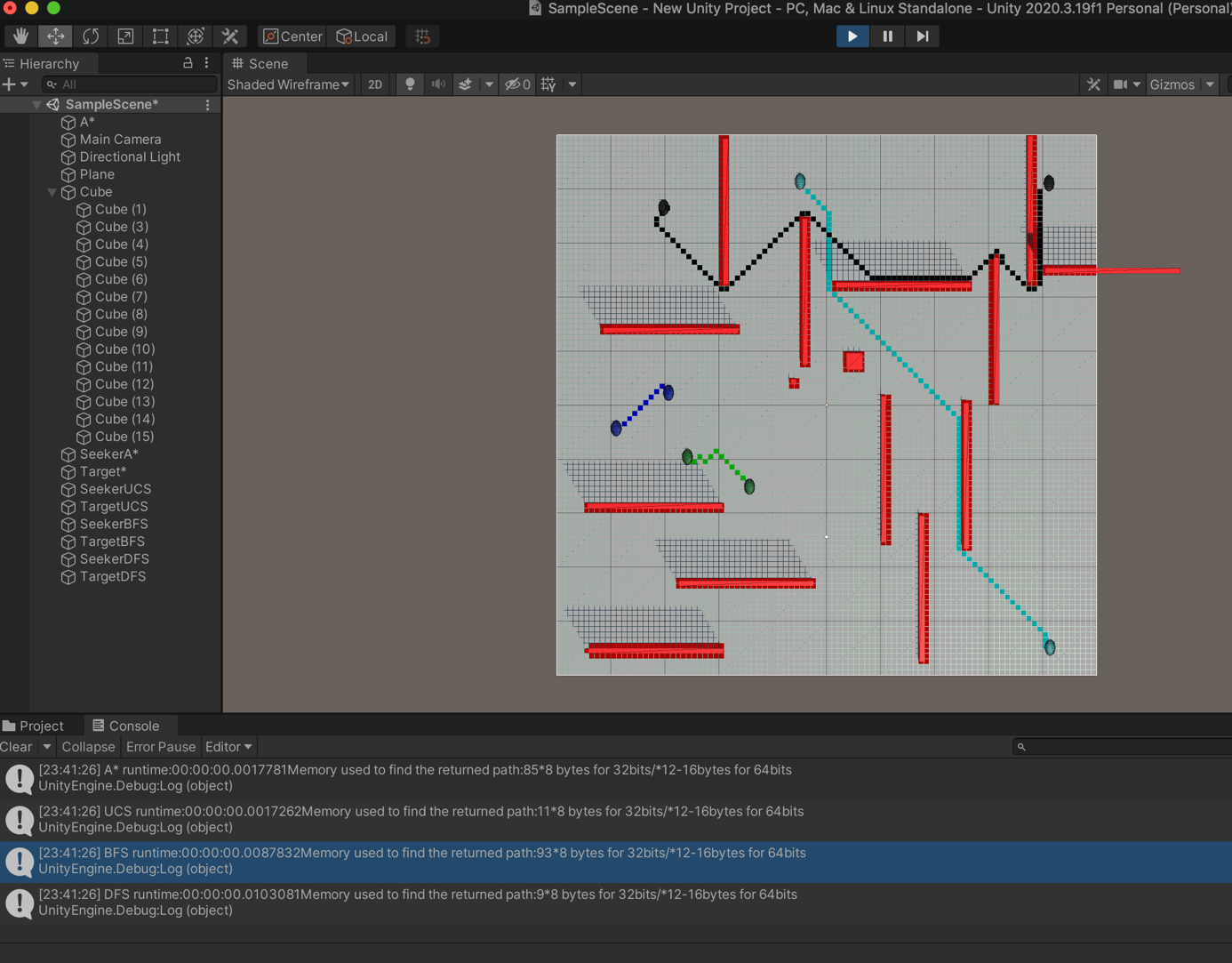




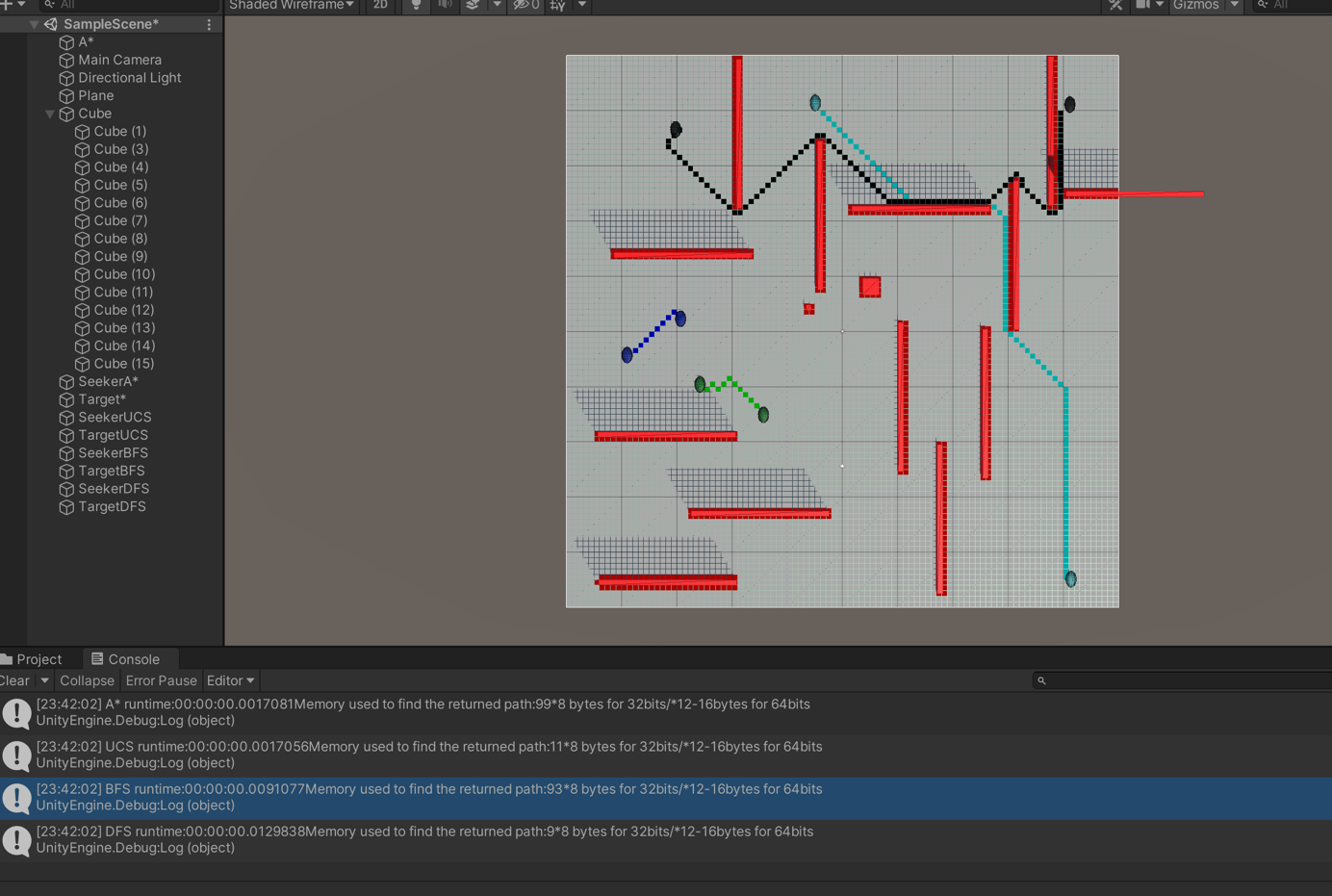


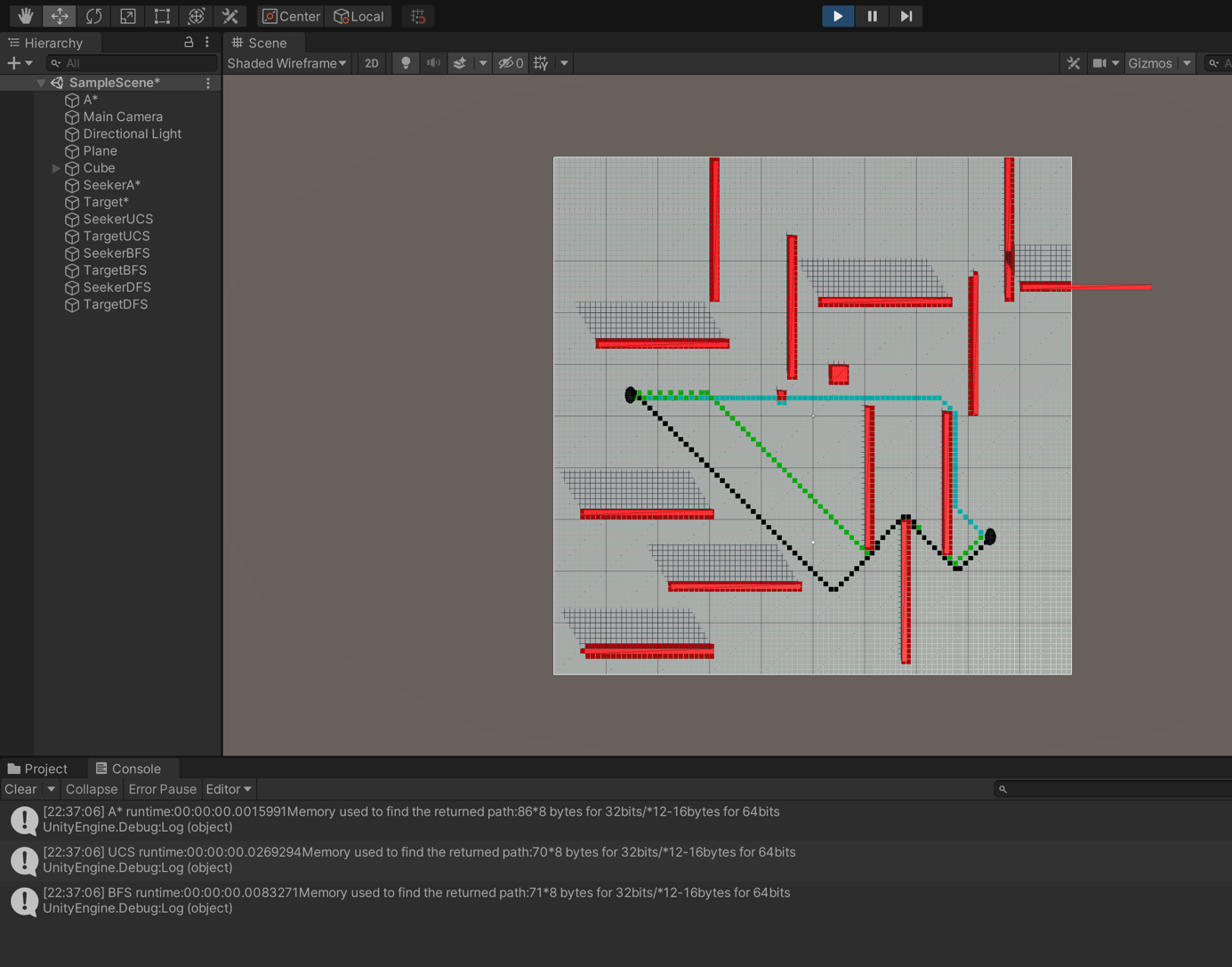






Here the cyan A\* algorithm is using double the original distance:

A\* with different heuristics seems to give a worse path if all directions are considered same

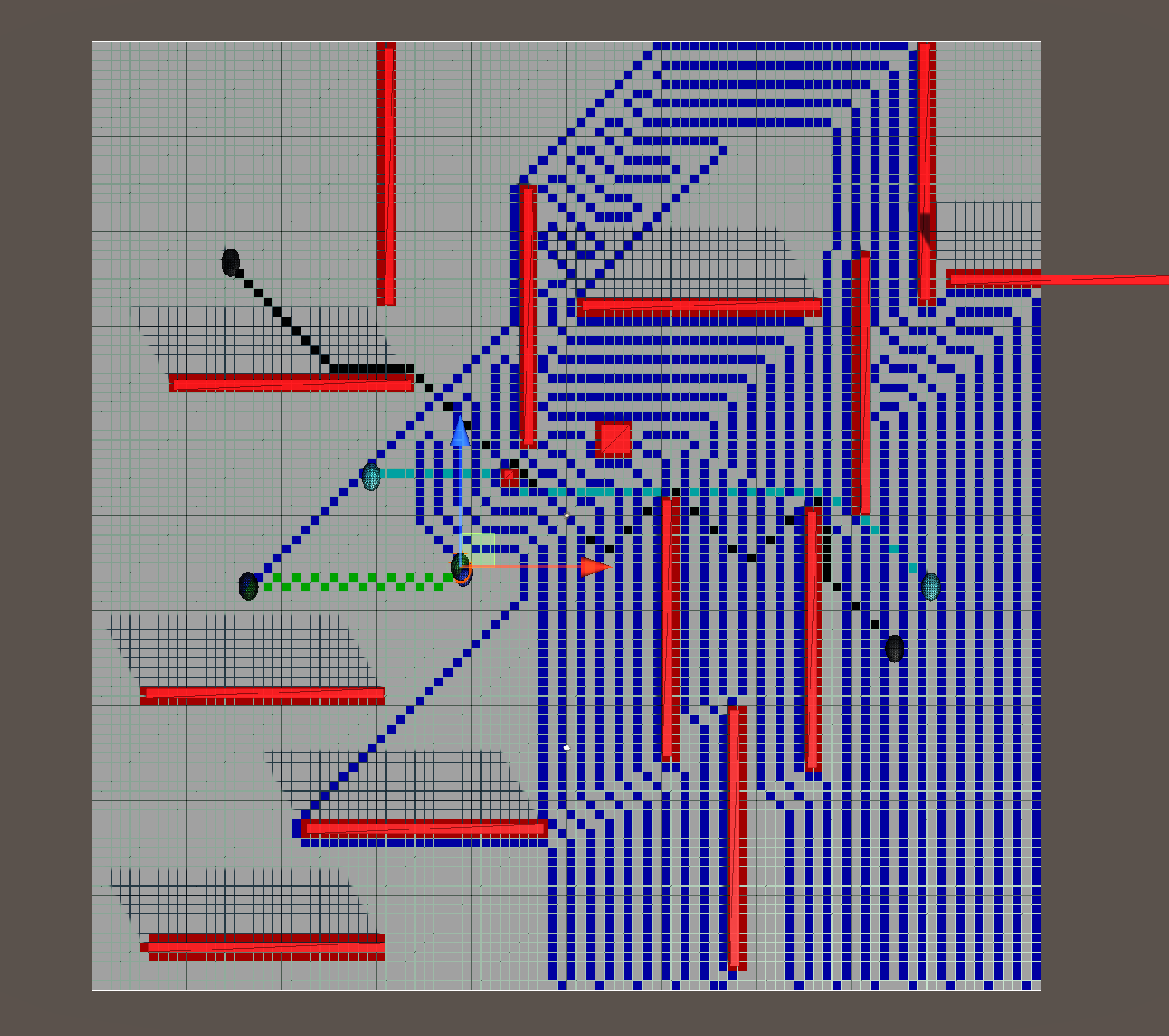


A\* with most optimal heuristic:

Une image contenant texte, horloge, capture d’écran

Description générée automatiquement

DFS terrible path due to exploring deep nodes:



Conclusion :

-When we change A\* heurestics the path changes depending on the new calculated distance, so we better keep the optimal distance used in sebastian’s video instead of using another path that can make the algorithm worse than the others or as good. Since A\* is the best algorithm used here.

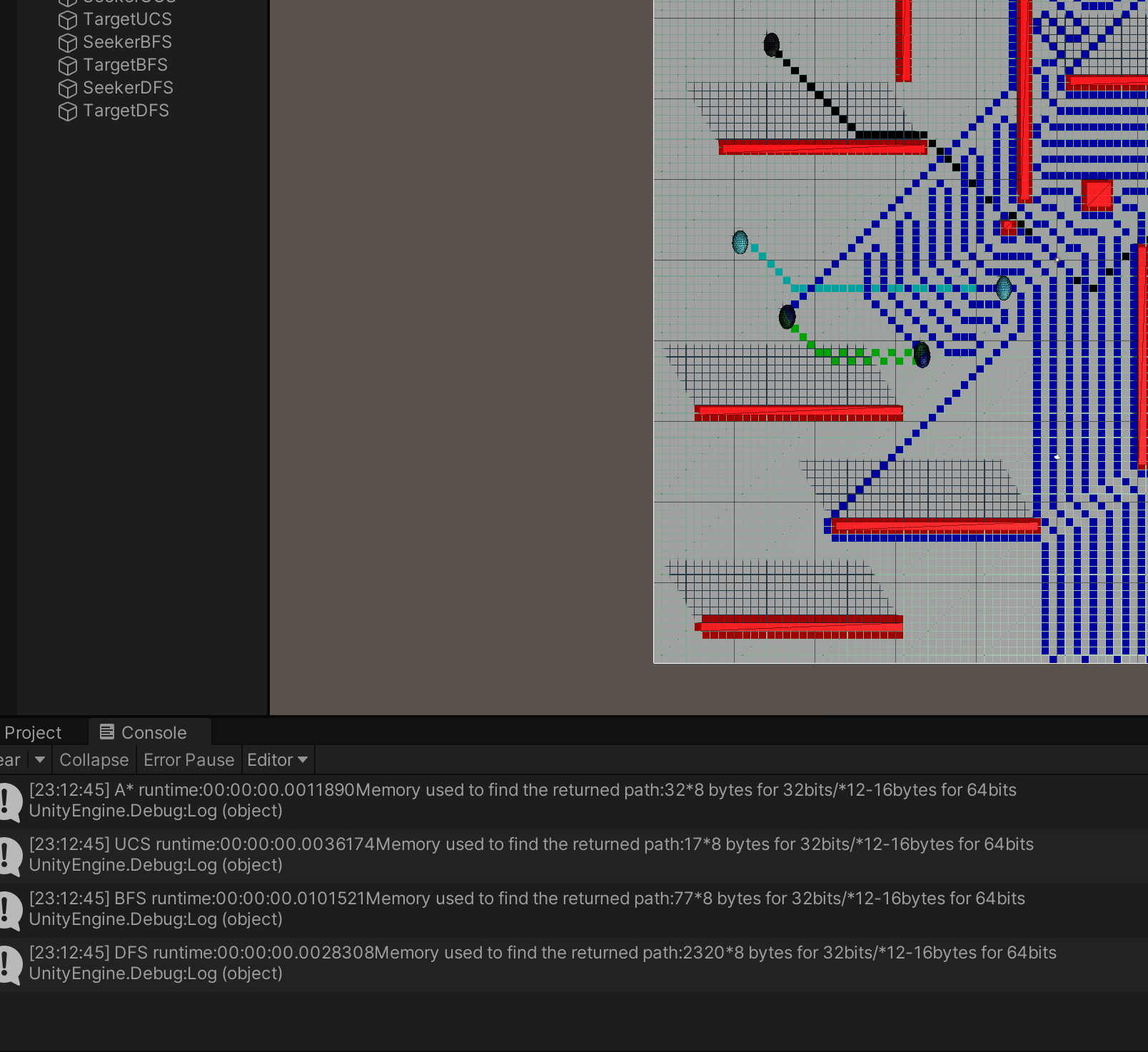
-DFS explores the deepest nodes until it finds the solution which explains the long blue lines that covered everything, it also uses a lot of memory for no reason other than exploring all nodes in the tree until it finds the goal and here the grid is open in all directions.

- BFS gives a good path but not the best

-UCS time can be same as DFS time and same as BFS space

-UCS can be better if A\* heuristics are not good

-A\* is the best when used with good heuristics (best path and best path and best memory usage)

-DFS and UCS can have almost same run time like here here BFS would have taken same space too

-UCS can have a better run time than DFS